Chương 2
Kiểu dữ liệu, biến, biểu thức

- Các kiểu dữ liệu cơ bản
- Hằng
- Biến
- Phép toán và biểu thức
- Hàm nhập xuất của C

Các kiểu dữ liệu cơ bản

- Các kiểu dữ liệu sơ cấp chuẩn trong C có thể được chia làm 2 dạng :
  - Kiểu số nguyên (integer, long)
  - Kiểu số thực (float, double)

Kiểu số nguyên

❖ Được dùng để lưu các giá trị nguyên hay còn gọi là kiểu đếm được.

- ***Kiểu số nguyên 1 byte (8 bits)***

STT	Kiểu dữ liệu	Miền giá trị (Domain)
1	unsigned char	Từ **0** đến **255** (*tương đương 256 ký tự trong bảng mã ASCII*)
2	**char**	Từ **-128** đến **127**

- ***Kiểu số nguyên 2 bytes (16 bits)***

STT	Kiểu dữ liệu	Miền giá trị (Domain)
1	enum	Từ -32,768 đến 32,767
2	unsigned int	Từ **0** đến **65,535**
3	short int	Từ -32,768 đến 32,767

- ***Kiểu số nguyên 4 byte (32 bits)***

STT	Kiểu dữ liệu	Miền giá trị (Domain)
1	**int**	Từ -32,768 đến 32,767
2	unsigned long	Từ **0** đến **4,294,967,295**
3	**long**	Từ **-2,147,483,648** đến **2,147,483,647**

Kiểu số thực

- Được dùng để lưu các số thực hay các số có dấu chấm thập phân

STT	Kiểu dữ liệu	Kích thước (Size)	Miền giá trị (Domain)
1	**float**	4 bytes	Từ **$3.4 * 10^{-38}$** đến **$3.4 * 10^{38}$**
2	**double**	8 bytes	Từ **$1.7 * 10^{-308}$** đến **$1.7 * 10^{308}$**
3	long double	10 bytes	Từ **$3.4 * 10^{-4932}$** đến **$1.1 * 10^{4932}$**

- Ngoài ra ta còn có kiểu dữ liệu void
  - *Mang ý nghĩa là kiểu rỗng không chứa giá trị gì cả*
  - ***Ví dụ:*** *void main(){ ....}*

Kiểu char

❖Kiểu kí tự

- Biểu diễn thông qua kiểu dữ liệu **char**
- Biểu diễn một ký tự thông qua bảng mã ASCII
- Hằng kí tự đặt trong cặp dấu ‘ ’

- Ví dụ:

```
char    s;
s = ‘d’;
```

❖Xâu (chuỗi) kí tự đặt trong cặp “ ”

- **Ví dụ:**

```
char    st[100];
st =“Chao cac ban!”;
```

Kiểu dữ liệu

❖ **Dùng sizeof():** Kích thước 1 kiểu có thể được xác định lúc chạy chương trình (runtime), dùng sizeof

- ***Ví dụ:***

sizeof(double) =>8(byte)

sizeof(long double)=>10(byte)

❖***Kiểu enum:*** *Nó cho phép ta định nghĩa 1 danh sách các bí danh (aliase) để trình bày các số nguyên.*

- ***Ví dụ:***

enum week { Mon=1, Tue, Wed, Thu, Fri Sat, Sun} days;

Định nghĩa kiểu với typedef

❖ Một khai báo có thêm tiền tố typedef sẽ định nghĩa một tên mới cho kiểu dữ liệu (đã có).

typedef KiểuDữLiệu tenMoi;

❖ Một tên được định nghĩa theo cách này được gọi là "định nghĩa kiểu".

❖Ví dụ:

```
typedef   long    SoNg32;
typedef   short  int   SoNg16;
typedef  char   KITU;
```

Hằng (Constant)

- Là đại lượng không đổi trong suốt quá trình thực thi chương trình
- Dùng toán tử **#define**

- Cú pháp:

#define <tên_hằng> <giá_trị_hằng>

- *Ví dụ:* #define MAX 100

- Biến hằng được định nghĩa nhờ từ khoá **const** với cú pháp như sau:

const <kiểu_dữ_liệu> <tên_biến> = <giá_trị>;

Ví dụ: const int MAX = 100;

Tên hằng số nên viết bằng chữ in HOA

Các loại hằng số

- Hằng số: Đó là các giá trị xác định, một hằng số có thể là *nguyên* (có kiểu int, long int…) hay *thực* (có kiểu float, double…).
- Hằng ký tự: Được đặt trong dấu nháy đơn ‘’. Ví dụ: 'A', 'a' tương ứng với giá trị nguyên 65, 97 trong bảng mã ASCII.
- Hằng chuỗi: Là tập hợp các ký tự được đặt trong cặp dấu nháy kép " ". Ví dụ: “Lap trinh C”
- ***Chú ý*:**
  - *“” : chuỗi rỗng - không có nội dung*
  - *Khi lưu trữ trong bộ nhớ, một chuỗi được kết thúc bằng ký tự NULL (‘\0’: mã Ascii là 0).*

Biến (variable)

- **Biến:** Là nơi lưu trữ dữ liệu trong bộ nhớ máy tính khi thực hiện chương trình, được đặt bởi một tên.
- Giá trị của biến có thể bị thay đổi.
- Mỗi biến chỉ có thể lưu một loại giá trị nhất định, tùy thuộc kiểu biến (kiểu dữ liệu).
- Phải khai báo biến trước khi sử dụng
- Cú pháp khai báo biến:

  <Kiểu dữ liệu> tênbiến;

  - ***Ví dụ:***

  int a; //Khai báo biến để lưu số nguyên tên a

  float dienTich; //Khai báo biến để lưu diện tích hình

Địa chỉ của các biến

- Là *địa chỉ (&) của các biến trong bộ nhớ* mà chúng ta cần nhập giá trị cho nó.
- Mỗi biến sẽ được lưu trữ tại một vị trí xác định trong ô nhớ, nếu kích thước của biến có nhiều byte thì máy tính sẽ cấp phát một dãy các byte liên tiếp nhau, địa chỉ của biến sẽ lưu byte đầu tiên trong dãy các byte này
- Cách lấy địa chỉ của biến: **&<tên biến>**

int anInt;

Khai báo và khởi tạo giá trị ban đầu cho biến

❖**Khai báo nhiều biến cùng kiểu**

<Kiểu dữ liệu> tênbiến1, tênbiến2, tênbiến3;

❑ Ví dụ:

```
int a, x, y;
```

❖**Khai báo và khởi tạo giá trị ban đầu cho biến**

<Kiểu dữ liệu> tênbiến = giá trị;

❑Ví dụ:

```
int a = 5;
float b = 5.4,  c = 9;
char ch = ‘n’;
```

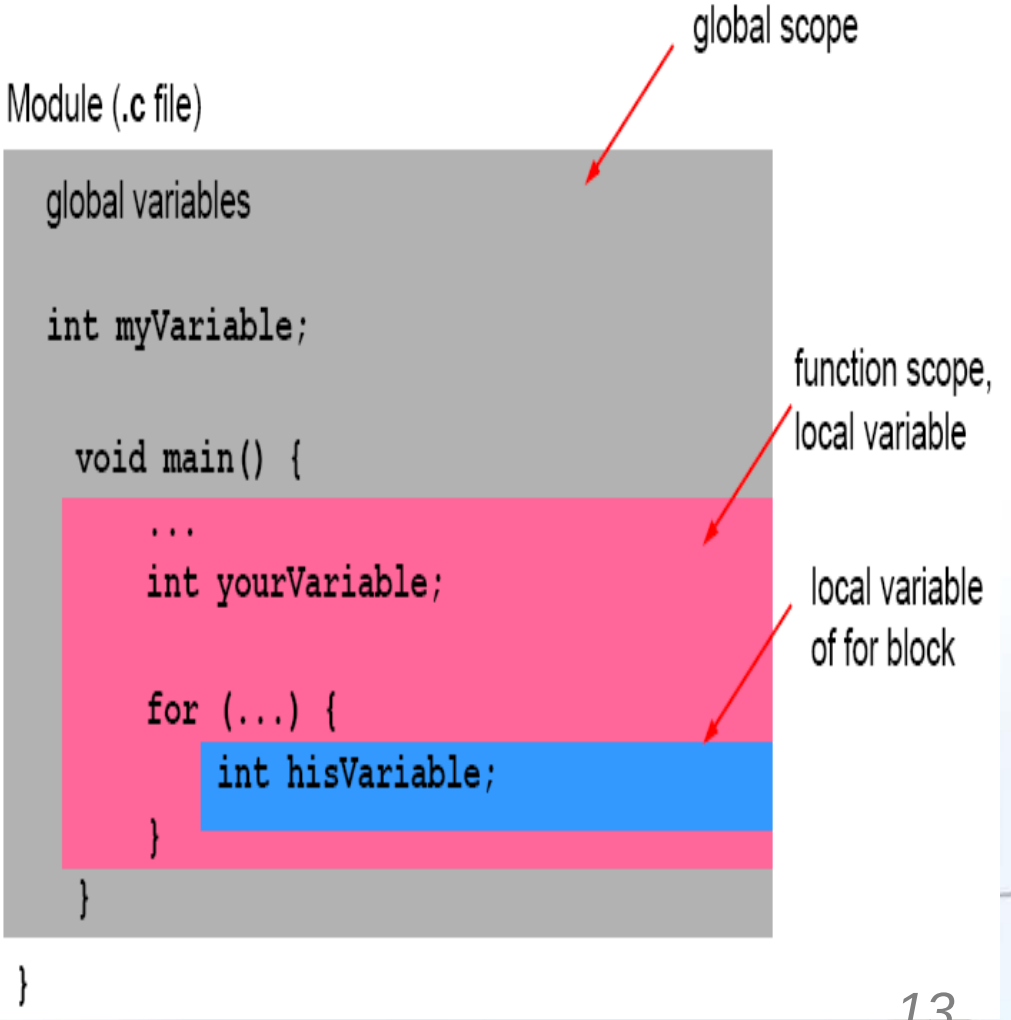
Tầm vực của biến

❖ Biến ngoài

- Được đặt bên ngoài tất cả các hàm
- Ảnh hưởng đến toàn bộ chương trình (**biến toàn cục**)

❖ Biến trong

- Được đặt bên trong hàm, chương trình chính hay một khối lệnh
- Ảnh hưởng đến hàm, chương trình hay khối lệnh chứa nó **(biến cục bộ).**



Tầm vực của biến

❖Biến tĩnh

- Biến tĩnh có kèm từ khóa **static** vào trước
- Tồn tại suốt chương trình
- Tầm vực sử dụng: toàn cục hoặc cục bộ
- Ví dụ: static int a;

Tầm vực của biến

❖ **Ví dụ:**

```
#include <stdio.h>
#include <conio.h>
int bienngoai;//khai báo biến ngoài
void main()
{
   int i=1, j=2;  //khai báo biến trong
   int bienngoai=3;
   printf("Gia tri i la %d\n",i);
   printf("Gia tri j la %d\n",j);
   printf("Gia tri bienngoai la %d\n",bienngoai);
   getch();
}
```

Phép toán và biểu thức

- Ví dụ:

  *(-b + sqrt(Delta))/(2*a)*

- Biểu thức là một sự kết hợp giữa
  - *Các toán tử (operator) và*
  - *Các toán hạng (operand)*
- Các loại toán tử trong C
  - *Toán tử số học*
  - *Toán tử quan hệ và logic*
  - *Toán tử Bitwise (thao tác trên bit)*
  - *Toán tử điều kiện*
  - *Toán tử con trỏ & và **
  - *Toán tử dấu phẩy*

Các toán tử số học

STT	PHÉP TOÁN	Ý NGHĨA	GHI CHÚ
PHÉP TOÁN SỐ HỌC			
1	+	Cộng	
2	-	Trừ	
3	*	Nhân	
4	/	Chia	Đối với 2 số nguyên thì kết quả là chia lấy phần nguyên
5	%	Chia lấy phần dư	Chỉ áp dụng cho 2 số nguyên
TOÁN TỬ TĂNG GIẢM			
6	++	Tăng 1	Nếu toán tử tăng/giảm đặt trước thì tăng/giảm trước rồi tính biểu thức hoặc ngược lại.
7	--	Giảm 1	

Các toán tử số học

+	Addition	x = 3+5 ⇒ 8
-	Subtraction	x = 5-3 ⇒ 2
*	Multiplication	x = 5*3 ⇒ 15
/	Division	x = 14/3 ⇒ 4 (integer division if both operands int)
%	Modulo	x = 14%3 ⇒ 2

- **Tăng và giảm (++ & --)**

++x hay x++ giống x = x + 1

--x hay x-- giống x = x – 1

Đâu là sự khác nhau?

- **Tuy nhiên:**

x = 10;

y = ++x;

Kết quả: y = 11, x=11

- **Còn:**

x = 10;

y = x++;

Kết quả: y = 10, x=11

Biểu thức Boolean (boolean expression)

- Không có kiểu Boolean (kiểu luận lý) rõ ràng trong C. Thay vào đó C dùng các giá trị nguyên để tượng trưng cho giá trị Boolean, với qui ước:

false	Giá trị 0
true	Bất kỳ giá trị nào ngoại trừ 0

- **Chú ý:** C dùng "=" cho phép gán, và dùng "= =" cho phép so sánh. Nó trả về 1 nếu bằng và 0 nếu không bằng

```
printf("%d\n", 1==2);   ⇒ 0
printf("%d\n", 1==1);   ⇒ 1
```

Các toán tử quan hệ và các toán tử Logic

- Các **phép so sánh** sau tạo ra các **biểu thức logic** có giá trị kiểu Boolean

Toán tử	Ý nghĩa
Các toán tử quan hệ	
>	Lớn hơn
>=	Lớn hơn hoặc bằng
<	Nhỏ hơn
<=	Nhỏ hơn hoặc bằng
= =	So sánh bằng
!=	So sánh khác nhau
Các toán tử Logic	
&&	AND
||	OR
!	NOT

Các toán tử quan hệ và các toán tử Logic

❖ **Ví dụ:**

Operator	Meaning	Example
==	equal	x == 3
!=	not equal	x != y
>	greater	4 > 3
<	less	x < 3
>=	greater or equal	x >= y
<=	less or equal	x <= y

- Các biểu thức logic trả về
  - **0** nếu **false (sai)**
  - **1** nếu **true (đúng)**

Các toán tử quan hệ và các toán tử Logic

❖ Bảng chân trị cho các toán tử Logic

p	q	p&&q	p||q	!p
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

❖ Thứ tự ưu tiên

Cao nhất:	!
	> >= < <=
	== !=
	&&
Thấp nhất:	||

❖ **Ví dụ:** (10>5)&&(10<9) => sai (0)

Các toán tử Bitwise (thao tác trên bit)

- Toán tử Bitwise giúp kiểm tra, gán hay thay đổi các bit thật sự trong 1 byte của word.
- Chỉ dùng cho kiểu char và int.

Toán tử	Ý nghĩa
&	AND
|	OR
^	XOR
~	NOT
>>	Dịch phải
<<	Dịch trái

Bảng chân trị của toán tử ^ (XOR)

p	q	p^q
0	0	0
0	1	1
1	0	1
1	1	0

Toán tử điều kiện

- Toán tử ? thực hiện như lệnh if-else.
- Cú pháp:

(ĐK)?<BT cho trường hợp đúng>:<BT cho trường hợp sai>

- **Ví dụ:** X = (10 > 9) ? 100 : 200;

  =>X=100

  X = (10 >15)? 100 : 200;

  =>X=200

Toán tử con trỏ & và *

- Ví dụ:

```
int *p;                    //con tro so nguyen
int count=5, x;
p = &count;
```

=>*Đặt vào biến p địa chỉ bộ nhớ của biến count*

- Toán tử * trả về nội dung của ô nhớ mà một con trỏ đang chỉ vào
  - *Ví dụ:*

```
x = *p;        // x=5
```

Toán tử dấu phẩy

❖ **Ví dụ**:

- *x = (y=3,y+1);*

Trước hết gán 3 cho y rồi gán 4 cho x.

❖ Được sử dụng để kết hợp các biểu thức lại với nhau.

❖ Bên trái của dấu (,) luôn được xem là kiểu void.

❖ Biểu thức bên phải trở thành giá trị của tổng các biểu thức được phân cách bởi dấu phẩy.

Độ ưu tiên phép toán

Toán tử	Độ ưu tiên	Trình tự kết hợp
`()  []  ->`	1	Từ trái qua phải
`!  ~  ++  --  sizeof`	2	Từ phải qua trái
`*  /  %`	3	Từ trái qua phải
`+  -`	4	Từ trái qua phải
`<<  >>`	5	Từ trái qua phải
`<  <=  >=  >`	6	Từ trái qua phải
`==  !=`	7	Từ trái qua phải
`&`	8	Từ trái qua phải
`|`	9	Từ trái qua phải
`^`	10	Từ trái qua phải
`&&`	11	Từ trái qua phải
`||`	12	Từ trái qua phải
`? :`	13	Từ phải qua trái
`=  +=  -=  *=  /=  %=`	14	Từ phải qua trái

Phép gán được viết gọn lại

Cú pháp: **x= x <phép toán> y;**

có thể được viết gọn lại (short form):

	short form	expanded form
+=	`x += y;`	`x = x + y;`
-=	`x -= y;`	`x = x - y;`
*=	`x *= y;`	`x = x * y;`
/=	`x /= y;`	`x = x / y;`
%=	`x %= y;`	`x = x % y;`

Các lỗi thường gặp khi viết chương trình

- Quên khai báo các biến sử dụng trong chương trình.
- Lưu một giá trị vào một biến nhưng không cùng kiểu dữ liệu với biến.
- Sử dụng biến trong một biểu thức khi nó chưa có giá trị. Lỗi này thì không được phát hiện bởi trình biên dịch, khi đó giá trị của biến là một giá trị bất kỳ và kết quả của biểu thức là vô nghĩa.

Phép gán

- Lệnh gán (assignment statement) dùng để gán giá trị của một biểu thức cho một biến.
- Cú pháp:

<Tên biến> = <biểu thức>;

Ý nghĩa: Gán giá trị cho 1 biến

- **Ví dụ:**

```
int main() {
    int x,y;
    x =10; // Gán hằng số 10 cho biến x
    y = 2*x; //Gán giá trị của biểu thức 2*x (=20) cho biến y
    return 0;
}
```

- Gán giá trị ngay tại lúc khai báo:

```
int x = 10, y=x;
```

Phép gán

- Kiểu của biểu thức và của biến phải giống nhau

```
int main() {
      int x,y;
      x = 10; //  Gán hằng số 10 cho biến x
      y = "Xin chao";
            //y có kiểu int, còn "Xin chao" có kiểu char*
      return 0;
}
```

Error: "*Cannot convert 'char *' to 'int'*"

Phép gán

- **Tự động chuyển kiểu:** Thường thì có sự chuyển đổi kiểu tự động nếu có thể. **Việc chuyển kiểu** được thực hiện từ toán hạng có kiểu "hẹp" sang kiểu "rộng" hơn
- Ví dụ:

```
int x, y; char ch;
y = 'd'; //y co kieu int, còn 'd' có kiểu char

float a = 3.34;
int b;
b = a;  //làm mất đi sự chính xác (loss of precision)
a = b;
```

Phép gán

❖Ép kiểu (casting type)

Cú pháp: (Tên kiểu) <Biểu thức>

Chuyển đổi kiểu của <Biểu thức> thành kiểu mới <Tên kiểu>

Ví dụ:

```
int   x;
double   y= 3.125;
y = (int) y;   // lúc này y có giá trị là ?
x = y;          //lúc này x=?
y = x/2;     //lúc này y có giá trị là ?
y = (double)x/2;  // lúc này y có giá trị là ?
y = x/2.0;      // lúc này y có giá trị là ?
```

Hàm nhập xuất của C

- **Thư viện <stdio.h>**
- **Xuất:**

printf(“hằng chuỗi”);

Vd: printf(“Xin chao cac ban”);

printf(“chuỗi định dạng”, đối số 1, đối số 2);

Chuỗi định dạng để qui định kiểu dữ liệu, cách biểu diễn, độ rộng, số chữ số thập phân, …

Vd: int a=5; float b=2.7;

printf(“Gia tri cua bien a=%d, b=%f“, a, b);

- **Nhập:**

scanf(“chuỗi định dạng”, &tên biến);

Vd: int x;

scanf(“%d”, &x);

Hàm nhập xuất của C

```
#include <stdio.h>
#include <conio.h>
void main ()
{
      clrscr(); //lệnh xóa trắng màn hình
       int age ;
       float size;

       printf ( “nhap tuoi“) ;
       scanf ( “%d”, &age) ;
       printf ( “chieu cao cua ban: “) ;
       scanf ( “%f”, &size) ;
       printf ( “ban %d tuoi va co chieu cao %0.2f  \n”,age, size);
       getch(); //dừng màn hình xem kết quả
}
```

➢ **%[số ký số][.số sau dấu phẩy]: Nhập số thực có tối đa <số ký số> tính cả dấu chấm, lấy ký số sau dấu thập phân**

Chuỗi định dạng

STT	KIỂU	GHI CHÚ	ĐỊNH DẠNG
KIỂU LIÊN TỤC (SỐ THỰC)			
1	float		%f
2	double		%lf
3	long double		%lf
KIỂU RỜI RẠC (SỐ NGUYÊN)			
1	char	Ký tự	%c
		Số nguyên	%d
2	unsigned char	Số nguyên dương	%d
3	int	Số nguyên	%d
4	unsigned int	Số nguyên dương	%u
5	long	Số nguyên	%ld
6	unsigned long	Số nguyên dương	%lu
7	char *	Chuỗi ký tự	%s

Xuất ký tự đặt biệt

- **Để biểu diễn ký tự đặc biệt bên trong chuỗi ta phải thêm dấu \ phía trước.**

Ký tự	Ý nghĩa	Ví dụ
\’	Xuất dấu nháy đơn	printf(“I\’m a student”); Kết quả: I’m a student
\”	Xuất dấu nháy đôi	printf(“ky tu \“dac biet\””); Kết quả: ky tu “dac biet”
\\	Xuất dấu chéo ngược “\”	printf(“ \\ ”); Kết quả: \
\0	Ký tự NULL	Dùng để gán ký tự kết thúc của chuỗi

Xuất ký tự đặt biệt

Ký tự	Ý nghĩa	Ví dụ
\t	Tab vào một đoạn ký tự trắng	printf(“ xyz\tzyx”); Kết quả: xyz zyx
\b	Xuất lùi về sau	printf(“ xyz\bzyx”); Kết quả: xyzyx
\n	Xuống dòng	printf(“ xyz\nzyx”); Kết quả: xyz zyx
\r	Về đầu dòng	printf(“ xyz\rzyx”); Kết quả: zyx

Sử dụng thư viện toán học include <math.h>

Phép Toán	**Diễn Giải**
abs(a)	Trị tuyệt đối của số nguyên
fabs(a)	Trị tuyệt đối của số thực
sqrt(a)	Căn bậc hai của số
pow(a,n)	a^n
M_PI	Số Pi= 3.1415